

WHAT WE'RE DOING:

Time to start playing with chips, or integrated circuits (ICs) as they like to be called. The external packaging of a chip can be very deceptive. For example, the chip on the Arduino board (a microcontroller) and the one we will use in this circuit (a shift register) look very similar but are in fact rather different. The price of the ATmega chip on the Arduino board is a few dollars while the 74HC595 is a couple dozen cents. It's a good introductory chip, and once you're comfortable playing around with it and its datasheet (available online <http://ardx.org/74HC595>) the world of chips will be your oyster. The shift register (also called a serial to parallel converter), will give you an additional 8 outputs (to control LEDs and the like) using only three Arduino pins. They can also be linked together to give you a nearly unlimited number of outputs using the same four pins. To use it you "clock in" the data and then lock it in (latch it). To do this you set the data pin to either HIGH or LOW, pulse the clock, then set the data pin again and pulse the clock repeating until you have shifted out 8 bits of data. Then you pulse the latch and the 8 bits are transferred to the shift registers pins. It sounds complicated but is really simple once you get the hang of it.

(for a more in depth look at how a shift register works visit: <http://ardx.org/SHIF>)

THE CIRCUIT:

Parts:



**CIRC-05
Breadboard Sheet
x1**



**2 Pin Header
x4**



**Shift Register
74HC595
x1**



Wire

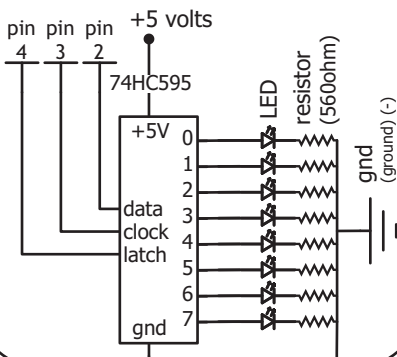


**Red LED
x8**



**560 Ohm Resistor
Green-Blue-Brown
x8**

Schematic



The Internet

.:download:.

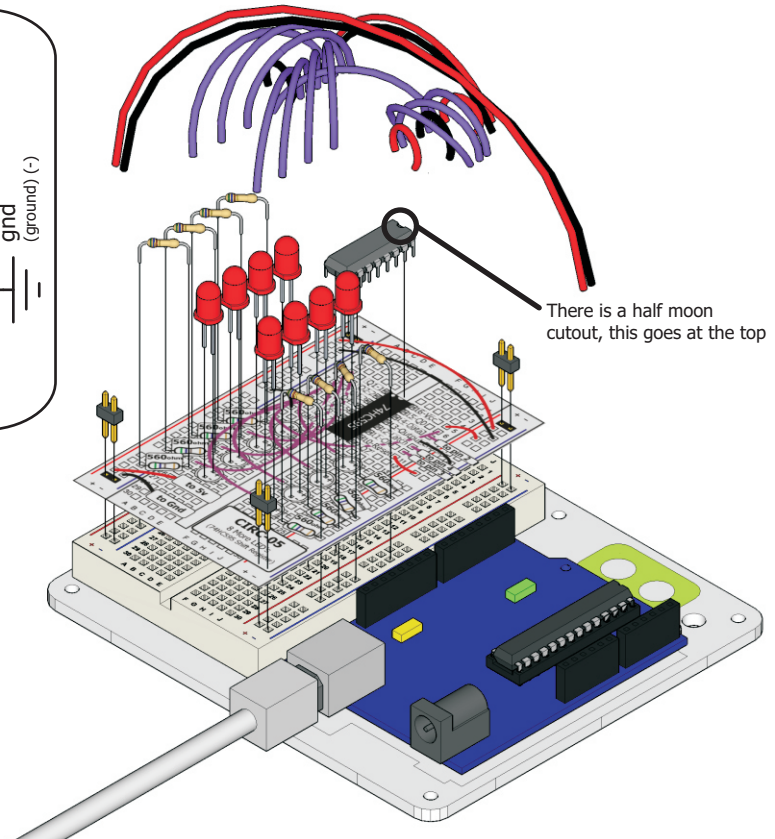
breadboard layout sheet

<http://ardx.org/BBLS05>

.:view:.

assembly video

<http://ardx.org/VIDE05>



CODE (no need to type everything in just click)**Download the Code from (<http://ardx.org/CODE05>)**

(copy the text and paste it into an empty Arduino Sketch)

```
//Pin Definitions
//The 74HC595 uses a protocol called SPI
//which has three pins
int data = 2;
int clock = 3;
int latch = 4;

void setup() //runs once
{
  pinMode(data, OUTPUT);
  pinMode(clock, OUTPUT);
  pinMode(latch, OUTPUT); }

void loop() // run over and over again
{
  int delayTime = 100;
  //delay between LED updates
  for(int i = 0; i < 256; i++){
    updateLEDs(i);
    delay(delayTime); }
}

/*
 * updateLEDs() - sends the LED states set
 * in value to the 74HC595 sequence
 */
void updateLEDs(int value){
```

```
digitalWrite(latch, LOW);
//Pulls the chips latch low
shiftOut(data, clock, MSBFIRST, value);
//Shifts out 8 bits to the shift register

digitalWrite(latch, HIGH);
//Pulls the latch high displaying the data
}
```

----- More Code Online -----

NOT WORKING? (3 things to try)**The Arduino's power LED goes out**

This happened to us a couple of times, it happens when the chip is inserted backwards. If you fix it quickly nothing will break.

Not Quite Working

Sorry to sound like a broken record but it is probably something as simple as a crossed wire.

Frustration?

Shoot us an e-mail, this circuit is both simple and complex at the same time. We want to hear about problems you have so we can address them in future editions.
help@oomlout.com

MAKING IT BETTER**Doing it the hard way:**

An Arduino makes rather complex actions very easy, shifting out data is one of these cases. However one of the nice features of an Arduino is you can make things as easy or difficult as you like. Let's try an example of this. In your loop switch the line:

```
updateLEDs(i) -> updateLEDsLong(i);
```

Upload the program and notice nothing has changed. If you look at the code you can see how we are communicating with the chip one bit at a time. (for more details <http://ardx.org/SPI>).

Controlling individual LEDs:

Time to start controlling the LEDs in a similar method as we did in CIRC02. As the eight LED states are stored in one byte (an 8 bit value) for details on how this works try <http://ardx.org/BINA>. An Arduino is very good at manipulating bits and there are an entire set of operators that help us out. Details on bitwise maths (<http://ardx.org/BITW>).

Our implementation.

```
Replace the loop() code with
int delayTime = 100; //the number of milliseconds
//to delay
```

```
for(int i = 0; i < 8; i++){ //between LED updates
  changeLED(i,ON);
  delay(delayTime);
}
for(int i = 0; i < 8; i++){
  changeLED(i,OFF);
  delay(delayTime);
}
```

Uploading this will cause the lights to light up one after another and then off in a similar manner. Check the code and wikipedia to see how it works, or shoot us an e-mail if you have questions.

More animations:

Now things get more interesting. If you look back to the code from CIRC02 (8 LED Fun) you see we change the LEDs using digitalWrite(led, state), this is the same format as the routine we wrote changeLED(led, state). You can use the animations you wrote for CIRC02 by copying the code into this sketch and changing all the digitalWrite()'s to changeLED()'s. Powerful? Very. (you'll also need to change a few other things but follow the compile errors and it works itself out).

MORE, MORE, MORE:

More details, where to buy more parts, where to ask more questions:

<http://ardx.org/CIRC05>